F009940
closed 07

AFRL-SR-AR-TR-07-0446

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	6/14/2007	FINAL PROJECT REPORT 2/1/2004-4/30/2007

4. TITLE AND SUBTITLE
Mixed-Initiative Development of Plans with Expressive Temporal Constraints

5. FUNDING NUMBERS
PA9550-04-1-0043

6. AUTHOR(S)
Martha E. Pollack

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The Regents of The University of Michigan
Div of Res Development & Administration
1058 Wolverine Tower
3000 South State Street
Ann Arbor, Michigan 48109

8. PERFORMING ORGANIZATION REPORT NUMBER F009940 047252 FINAL

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. Robert Herklotz
USAF, AFRL (AFOSR PKR2)
Air Force Office of Scientific Research
875 North Randolph Street, Room 3112
Arlington, VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The overall objective of this project was the development of methods for efficiently finding solutions to problems that consist of sets of expressive constraints, including both overconstrained and underconstrained situations. In either case, constraint management is necessary: in an overconstrained situation, it is necessary to determine how best to relax existing constraints so that a solution can be computed, while in an underconstrained situation, it is necessary to select good (or, if time permits, optimal) solutions from amongst the alternatives. We developed novel and highly efficient solutions using two classes of techniques: fully automatic techniques that manage constraint sets given explicit preference functions, and mixed initiative techniques that allow interactive control by a human user to manage constraint sets, and we evaluated these on both synthetic and real-world data sets.

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	unclassified	unclassified	

Mixed-Initiative Development of Plans with Expressive Temporal Constraints
Final Report
AFOSR Contract Number FA9550-04-1-0043

Martha E. Pollack
Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109, USA
{pollackm}@umich.edu

June 14, 2007

1 Executive Summary

1.1 Project Overview

The overall objective of this project was the development of methods for efficiently finding solutions to problems that consist of sets of expressive constraints, including both overconstrained and underconstrained situations. In either case, constraint management is necessary: in an overconstrained situation, it is necessary to determine how best to relax existing constraints so that a solution can be computed, while in an underconstrained situation, it is necessary to select good (or, if time permits, optimal) solutions from amongst the alternatives. We developed novel and highly efficient solutions using two classes of techniques: fully automatic techniques that manage constraint sets given explicit preference functions, and mixed initiative techniques that allow interactive control by a human user to manage constraint sets. The solutions we developed are broadly applicable to challenges that arise in the domains of cybersecurity and cyberdefense. However, because of the lack of accessible test cases in these domains, we validated our techniques using both systematic but abstract datasets, and large real constraint problems taken from another domain: floorplan legalization in circuit design. An added advantage of working in this domain was the demonstration that our methods, which were developed in the context of temporal reasoning, also apply to problems of spatial reasoning.

1.2 Key Results

Amongst the key results of this project are the following:

- Development of two approaches to finding solutions to overconstrained Disjunctive Temporal Problems (DTPs) that maximize the number of satisfied constraints. The first uses the meta-level transformation of standard (hard-constraint only) DTP solvers, to achieve a 3 order-of-magnitude efficiency gain relative to a baseline technique [15], and a second approach that uses local search and produces an additional order-of-magnitude speed-up [8].
- Development of a method for extracting minimum conflict sets from overconstrained DTPs, which can be used to support mixed-initiative resolutions by providing a user with an indication of the source of conflicts [9].
- Development of techniques for finding optimal solutions to DTPs, where the optimality criterion is maximin (maximize the minimal value assigned to any constraint [17]. Our algorithm adds only polynomial time complexity relative to the cost of solving the reduced problem in which all the soft constraints are treated as if they were hard.
- Development of techniques for finding optimal solutions to STPs, where the optimality criterion is utilitarian (maximize the sum of the values assigned to each constraint [14], and for finding optimal solutions to DTPs, with the same optimality criterion [7,3]. The STP results were the initial results on this problem, and thus establish baseline conditions; the DTP results provide a 3 order-of-magnitude speed-up relative to the state-of-the-art.
- Identification of a new type of uncertainty that was previously inexpressible in the STP and DTP formalisms, and generalization of the formalisms to handle this [1,2].
- Validation of the effectiveness of the techniques developed on "real" problems from the circuit-layout domain (results in the earlier work were based on synthetic data sets) [4,5].

2 Background

Note: Throughout the following sections, we provide a brief discussion of the major results of the project. References are limited to our own publications. More detailed discussion, including complete bibliographic citations, can be found in our publications.

The starting point for the work done in this project is the Disjunctive Temporal Problem (DTP), which are a generalization of the Simple Temporal Problem (STP).An STP is a pair $\langle X, C \rangle$, where the elements $X_i \in X$ designate time-points, and C is a set of binary temporal constraints of the following form:

$$X_j - X_i \in [a_{ij}, b_{ij}].$$

A *solution* to an STP is an assignment of values to time-points that satisfies all constraints. An STP is said to be *consistent* if at least one solution exists. *Consistency-checking* in an STP can be cast as an all-pairs shortest path problem in the corresponding network: the STP is consistent iff there are no negative cycles in the all-pairs graph.

This check can be performed in $O(|X|^3)$ time. A by-product of this check is the *minimal network*, which is the tightest representation of the STP that still contains all solutions present in the original network. A single solution can be extracted from the minimal network in $O(|X|^2)$ time.

A Disjunctive Temporal Problem (DTP) is a pair $\langle X, C \rangle$, where each element of C is a disjunction of STP constraints as in the following:

$$(X_{j1} - X_{i1} \in [a_{i1j1}, b_{i1j1}]) \vee (X_{j2} - X_{i2} \in [a_{i2j2}, b_{i2j2}]) \vee$$

$$\cdots \vee (X_{jn} - X_{in} \in [a_{injn}, b_{injn}]).$$

To satisfy a DTP constraint, only one of its disjuncts needs to be satisfied. An assignment that satisfies at least one disjunct in each constraint is a solution to a DTP.

A key construct used for solving a DTP is a *component STP*, which is created by selecting one disjunct from every DTP constraint. A given DTP has a solution if and only if one of its component STPs does. Therefore, the search for a solution to a DTP can be carried out by searching for a solution in each of its component STPs.

Finding a solution to an STP requires only polynomial time, but a DTP can have an exponential number of component STPs. Fortunately, backtracking search and pruning techniques make this approach practical in many cases: the component STP is built up one disjunct at a time, backtracking if the current set of disjuncts are inconsistent.

The search through a DTP's component STPs is often cast as a search through a meta-CSP derived from the DTP. Each variable in the meta-CSP corresponds a disjunctive constraint in the DTP; each disjunct in the disjunctive constraint is represented by a single value in the meta-CSP variable. Assigning a meta-CSP variable to a particular value is therefore equivalent to choosing a particular disjunct from a constraint to include in a component STP. Consequently, a set of assigned variables in the meta-CSP defines an STP composed of all disjuncts implied by the assignments.

3 Overconstrained DTPs

To deal with the problem of DTPs that are not consistent, i.e., allow no assignments that satisfy all the constraints, we explored two approaches. In the first, we built on partial constraint satisfaction (PCS) techniques, which find partial solutions to a given CSP by seleting a subset of constraints to relax. Often, this is done with the objective being to minimize the total number of weakened constraints, and we adopted this goal in our work as well. Where previous work had applied partial constraint satisfaction techniques to finite-domain CSPs, we showed how to adoptthe problem to the meta-CSP model of DTP solving, in which the constraints of the original problem become the variables of a meta-level problem.

Our algorithm, which was called Maxilitis, derives from the simple branch and bound algorithm given in Figure 1. On top of this framework, we add each of the additional meta-level pruning techniques that we had exploited in our earlier work on the Epilitis DTP solver, with the exception of no-good recording, whose general applicability to partial constraint satisfaction is a focus of future work. More specifically, we demonstrated how to incorporate both general pruning techniques such as forward

Partially-Solve-DTP(A, U, *distance*, *upperbound*)
1. If (*distance* $\geq$ *upperbound*) return
2. If ($U = \emptyset$)
3. *best-solution-so-far* $\leftarrow A$
4. *upperbound* $\leftarrow$ *distance*
5. return
6. EndIf
7. $C \leftarrow$ *select-variable*(U), $U' \leftarrow U - \{C\}$
8. For each value c of $d(C)$
9. $A' \leftarrow A \cup \{C \leftarrow c\}$
10. If **consistent**(A')
11. **Partially-Solve-DTP**(A', U', *distance*, *upperbound*)
12. EndIf
13. EndFor
14. $A' \leftarrow A \cup \{C \leftarrow \epsilon\}$
15. **Partially-Solve-DTP**(A', U', *distance* + 1, *upperbound*)

Figure 1: A simple partial constraint satisfaction algorithm for DTPs

checking and conflict-directed backjumping as well as techniques specific to temporal CSPs, such as removal of subsumed variables and semantic branching.

Experimental results, given in [15] demonstrated that the computation time required to partially solve DTP problems is dramatically reduced by applying the pruning techniques of removal of subsumed variables and semantic branching. They also demonstrated that by executing our solver multiple times with incrementally increasing upper bounds on the solution distance, the performance can be improved even more, at the cost of giving up the algorithm's anytime quality. Unfortunately, despite these speedups, exact partial constraint satisfaction appears to become intractable when the number of constraints or variables in the problem becomes substantially large.

In response to this latter issue, we developed an alternative technique, which is based on local search. The technique differs markedly from previous work on DTPs, as it operates within the total assignment space of the underlying CSP rather than the partial assignment space of the related meta-CSP. While search in the partial assignment space of the meta-CSP is common to most systematic methods for solving DTPs, it is less attractive with local search for several reasons. First, systematic methods work within a backtracking tree, where disjuncts are removed in the order in which they were added.m One technique that is commonly used in DTP solving, *incremental full-path consistency*, exploits this property by maintaining a stack of the variable assignments made during the search, and using it to cheaply update path dependencies during backtracking. Local search requires the ability to modify arbitrary values in the partial assignment, not necessarily respecting the order in which they were originally assigned; thus it cannot exploit the incremental approach. Second, several of the powerful pruning techniques used by DTP solvers have no meaning outside the context of a systematic search tree. For example, semantic branching is able to acquire additional network constraints by exhaustively exploring particular assignments of disjuncts to constraints. In local search, no such exhaustive search is performed, and consequently

it is hard to imagine how to adapt this mechanism. Finally, in local search, the number of neighbors for partial assignment will typically be much larger than the number of successor in systematic search, thus making evaluation of alternatives prohibitively expensive.

The alternative approach is to perform search is the space of total assignments to the time points in the original CSP. However, some key issues arise, which we addressed in our work. One of the most important is the question of how to define the neighbors of an assignment,without creating an infinite search space. The key to reducing the size of the search space is to note that if we hold fixed the values of all the variables but one in a given problem, then only a small set of new values for the selected variable are significant. Specifically, we only need to consider those values for which the *slack* of some disjunct of some constraint becomes zero–in other words, those values which cause an inequality to become a strict equality. This is somewhat similar to a pivot step in the simplex method for linear programming, which maintains a basic feasible solution that corresponds to an "active system" of constraints in the LP.

Using this key insight, as well as an approach to neighbor selection and tabu moves, we developed the Localitis algorithm, shown in Figure 2. Experimental results, presented in [8], demonstrated that the local-search approach requires computation time that is significantly reduced in comparison to traditional branch-and-bound algorithms for performing partial constraint satisfaction.

4 Minimum Conflict Sets

An alternative approach to solving overconstrained sets of constraints is to involve a user in a mixed-initiative process. Towards this end, it is important to able to identify minimal sets of constraints that are unsatisfiable, so that the user can determine what sorts of relaxations are appropriate. We thus developed a technique for efficiently identifying sets of conflicting constraints in an overconstrained problem. Our approach is particularly well-suited to temporal problems, in which conflicts among constraints can be resolved by weakening, rather than completely abandoning, constraints.

We built on our previous work on identifying Minimally Unsatisfiable Subsets of constraints (MUSes). Given a set of constraints C, an MUS of C is a subset of C that is (1) unsatisfiable and (2) minimal, in the sense that removing any one of its elements makes the rest of the MUS satisfiable. Each MUS thus provides information about a conflict that must be addressed to solve the given CSP. In general, an arbitrary CSP may contain multiple MUSes, and all of them must be resolved by constraint relaxation before the CSP can be solved. Identifying the MUSes of a CSP makes it possible to reason about how to weaken conflicting constraints to make a solution feasible.

Our techniques for extracting MUSes are derived from a deep relationship between maximal satisfiability and minimal unsatisfiability. The Maximal Constraint Satisfaction problem (Max-CSP) is an optimization problem on a constraint system C that has the goal of finding an assignment to the variables of C that satisfies as many constraints as possible.

While Max-CSP is defined in terms of the cardinality of a satisfiable subset of constraints, the definition can be relaxed to have *inaugmentability* as the goal instead.

Localitis(DTP D)
1. *Best-Assign* $\leftarrow$ *Assign* $\leftarrow$ Maxilitis-First-Path(D)
2. For $it = 1$ to *max-steps*
3. *min_cost* $\leftarrow \infty$, *moves* $\leftarrow \emptyset$
4. For each disjunct d: $x - y \leq b$
5. If x not tabu
6. *move* $\leftarrow (x \leftarrow y + b)$
7. If Sat(*Assign*) $\neq$ Sat(*Assign/move*)
8. If cost(*Assign/move* $<$ *min_cost*)
9. *moves* $\leftarrow \emptyset$
10. *min_cost* $\leftarrow$ cost(*Assign/move*)
11. EndIf
12. If cost(*Assign/move* $=$ *min_cost*)
13. *moves* $\leftarrow$ *moves* $\cup$ {*move*}
14. EndIf
15. EndIf
16. EndIf
17. Repeat lines 5 – 16 for y (i.e., $y \leftarrow x - b$)
18. EndFor
19. *new-move* $\leftarrow$ Random-Member(*moves*)
20. *Assign* $\leftarrow$ *Assign/new-move*
21. If cost(*Assign*) $<$ cost(*Best-Assign*)
22. *Best-Assign* $\leftarrow$ *Assign*
23. EndIf
24. Update tabu count for the time point in *new-move*
25. EndFor
26. return *Best-Assign*

Figure 2: Localitis, a local search algorithm for DTPs

Thus, while we can define Max-CSP(C) as $\{m \subseteq C : |m| \text{ is maximal}, m \text{ is satisfiable}\}$, we can define a new problem, *Maximally Satisfiable Subset* (MSS). The definition of the set of MSSes follows, with the set of MUSes defined similarly for comparison:

$$\text{MSSes}(C) = \left\{ \begin{array}{l} m \subseteq C : m \text{ is satisfiable, and} \\ \forall c \in (C \backslash m),\ m \cup \{c\} \text{ is unsatisfiable} \end{array} \right\}$$

$$\text{MUSes}(C) = \left\{ \begin{array}{l} m \subseteq C : m \text{ is unsatisfiable, and} \\ \forall c \in m,\ m \backslash \{c\} \text{ is satisfiable} \end{array} \right\}$$

In general, given any MSS, the set of constraints not included in that MSS provides an irreducible "fix" for the original infeasible system; removing these constraints makes it satisfiable. Therefore, we define a "CoMSS" as the complement of an MSS, and the set CoMSSes(C) as:

$$\text{CoMSSes}(C) = \{m \subseteq C : (C \backslash m) \in \text{MSSes}(C)\}$$

Our algorithm depends on these interelationships. In brief, we use a serial decomposition of the task. First we find all of the CoMSSes of a DTP using an algorithm

that borrows heavily from Maxilitis, discussed above; this algorithm could be readily generalized for other types of constraints. Second, we use techniques for extracting MUSes that operate completely independently of how the CoMSSes were generated and the types of constraints involved. The two phases combine to form a solver, which we name Musilitis, that is capable of diagnosing the infeasibility of any given DTP. The solver is both sound, in that the sets of constraints it returns are all MUSes, and complete, in that it will find all MUSes of the given constraint system.

We benchmarked both phases of the MUS generation process (finding CoMSSes(C) and extracting MUSes(C) from it) using DTPs created by a random DTP generator used in testing previous DTP solvers. We collected performance data and analyzed the sets of CoMSSes and MUSes for DTPs over a range of DTP generator parameters. Experimental results, given in [9], show excellent anytime properties. In one typical highly-constrained instance, our algorithm generated more than 70,000 MUSes in the first minute, and the rate then gradually slowed. This property could be exploited in a system that interleaves MUS identification with constraint relaxation. Generally, if the constraint system is highly overconstrained, resolving one MUS is likely to resolve many others, since the same constraints are likely to appear in a large number of MUSes.

5 Underconstrained Problems

A large focus of the project has been on the development of techniques for solving temporal constraint problems with soft constraints (or preferences). In this case, one needs to consider the criterion that will be used for determining the quality of solution. We explored two different optimality criteria, both of which have received attention in the prior literature: maximin optimality and utilitarian optimality. The former equates the value of an overall solution with the minimum value assigned to any individual constraint, and attempts to maximize that value. The latter equates the value of an overall solution with the sum of the values assigned to individual constraints, and agin tries to maximize the overall value.

While there had been prior work on finding maximin optimal solutions to Simple Temporal Problems (STPs) (where the criterion was called "Weakest Link Optimality"), we addressed the question of how to do this for DTPs. Our algorithm borrows concepts from previous algorithms for solving STPs with Preferences (STPPs) and TCSPs, in both cases using techniques for projecting and solving component subproblems. We demonstrate that the added expressivity provided by preferences is computationally inexpensive. Specifically, we show that our algorithm not only falls into the same complexity class as the corresponding hard-constraint algorithms, but that the added cost of handling such preferences is worst case polynomial in the number of time points in the network.

More specifically, the algorithm has two phases. In the first, the preference values in the DTPP are discretized and a set of DTPs is projected from the DTPP–one DTP for each preference value. This is done in a manner similar to that of the STPP algorithm. In the second phase, an attempt is made to solve each DTP by searching for a consistent component STP. The goal is to find the DTP with highest preference value

that is consistent.

Significant efficiency gains are obtained by exploiting the *Upward Inconsistency Property*: If a component STP of DTPq is inconsistent, then any related component STP of DTPp for $p > q$ will also be inconsistent.

This property holds because the higher-level STP constraints are tighter than the lower-level STP constraints belonging to the same family. If no solution exists in an STP, no tighter version of that STP will contain a solution. The upward inconsistency property allows us to prune away a large part of the search space each time an inconsistent STP is found. For example, if the algorithm finds an inconsistent STP at preference level 1, then it can prune away all related higher-level STPs.

A complexity analysis in [17] demonstrates that our approach does not add significant complexity relative to the cost of solving a DTP that contains only hard constraints.

In follow-on work, we explored the alternative notion of utilitarian optimality. In [14] presents a novel algorithm for finding utilitarian optimal solutions to Simple Temporal Problems with Preferences (STPPs) called the Greedy Anytime Partition algorithm for STPPs (GAPS). GAPS is an iterative algorithm that does not restrict preference functions and exhibits appealing properties that make it suitable for planning and scheduling:

Anytime Finds solutions that average over 80% of optimal after a single iteration, and up to 99% of optimal after m^2 iterations, where m is the number of constraints. Also performs comparably to a previous algorithm for STPPs that handles only convex preference functions.

Complete Finds optimal solution in time that compares favorably to a branch-and-bound algorithm.

Memory-boundable Allows caller to define trade-off between space and anytime performance.

GAPS borrows the idea of projecting preference functions onto hard STP constraints, allowing standard, polynomial-time algorithms for solving STPs to be leveraged. Rather than operating directly on soft constraints, i.e., constraints with a preference function, our algorithm first converts them into *preference projections*, which represent each soft constraint using a set of hard STP constraints, i.e., those without preference functions.

GAPS relies on another algorithm, STPP_Greedy, a simple algorithm for quickly finding a single high-quality solution to an STPP. The algorithm searches the space of component STPs, starting with the lowest-valued, $ROOT$, and repeatedly improving its value by replacing a single constraint with one of its children. The main function, `replaceAConstraint`, picks a constraint, replaces it with one of its children, and returns the child's identifier or a failure flag if no replacement is possible. If the replacement by a child leads to an inconsistent component STP, a greedy decision is made: the constraint is restored, marked as "finished" and never again chosen. STPP_Greedy is illustrated in Figure 3.

GAPS then starts by running STPP_Greedy to find a greedy solution G (i.e. a component STP). Then, GAPS uses G to partition the entire STPP search space into

n+1 smaller subproblems, n of which will be placed in a priority queue to be recursively solved later with GAPS, and one of which will be pruned. After this first iteration, GAPS repeats these steps on a subproblem removed from the queue, keeping track of the best solution found by each call to STPP_Greedy. When the queue empties, the best solution will be the optimal solution. The interesting element of GAPS is how it partitions the STPP search space using the component STP G; detals of how to do this efficiently are provided in [14].

As noted above, we demonstrated that GAPS is complete and memory-bounded and we provided experimental results showing that (1) a single iteration produces high-quality solutions, (2) multiple iterations, bounded by the square of the number of constraints, produce near-optimal solutions.

Subsequently to our development of algorithms for finding utilitarian optimal solutions to STPs, we moved on to doing the same thing for DTPs. Our first effort [7], embedded the approach within a Mixed Logigal Integer Linear Programming (MLLP) framework involving two types of constraints: logical constraints over Boolean variables, and Unit- Two-Variable-Per-inequality (UTVPI) integer constraints, while the second [3] departed from the SAT encoding and instead introduced a new formalism, the Valued DTP (VDTP). In contrast to the traditional semiring-based formalism that annotates legal tuples of a constraint with preferences, the VDTP assigns elementary costs to the constraints themselves. While this reformulation provides no increase in expressive power, it drastically simplifies the computational difficulties related to temporal optimization, since (unlike algorithms for solving finite-domain CSPs) search strategies for disjunctive temporal reasoning rarely invoke object-level assignments directly.

We proved that the VDTP can express the same set of utilitarian optimal solutions as the DTPP with piecewise-constant preference functions, and we develoedp a method for achieving *weighted constraint satisfaction* within a meta-CSP search space that has traditionally been used to solve DTPs without preferences. This allows us to directly incorporate techniques developed in previous decision-based DTP literature in order to make preferential optimization particularly efficient. Our algorithm is shown in Figure 4. The input variable A is the current set of assignments to meta-variables, and is

STPP_Greedy($ROOT$)
1. IF $ROOT$ is inconsistent, RETURN Ø
2. cSTP ← $ROOT$
3. DO
4. $\langle k, l, i \rangle$ ← `replaceAConstraint`(cSTP)
5. IF $k \neq$ -1 AND cSTP is inconsistent
6. `markAsFinished`($C_{\langle k,l,i \rangle}$)
7. cSTP[k] ← `parentOf`($C_{\langle k,l,i \rangle}$)
8. END IF
9. WHILE $k \neq$ -1

RETURN cSTP

Figure 3: The STPP_Greedy algorithm.

```
Solve-VDTP(A, U, cost, upperbound)
If (cost ≥ upperbound) return
If (U = ∅)
    best-solution-so-far ← A
    upperbound ← cost
    return
EndIf
C_i ← select-variable(U), U' ← U − {C_i}
For each disjunct c_ij of D(C_i)
    A' ← A ∪ {C_i ← c_ij}
    If (consistent(A'))
        Solve-VDTP(A', U', cost, upperbound)
    EndIf
EndFor
A' ← A ∪ {C_i ← ε}
Solve-VDTP(A', U', cost + w_i, upperbound)
```

Figure 4: A branch-and-bound algorithm for solving VDTPs

initially $\oslash$; variable U is the set of unassigned meta-variables (initially the entire set C); *cost* is the total weighted sum of violated constraints (initially zero); and *upperbound* is the stored cost of the best solution found so far (initially set to ∞).

This algorithm, which takes an approach similar to that our Maxilitis algorithm discussed above, resembles the meta-CSP backtracking search commonly used for solving traditional DTPs with two notable differences. First, backtracking occurs only when the combined weight of the violated constraints (*cost*) equals or exceeds that of the current best solution (*upperbound*); in a standard DTP solver, backtracking would occur whenever *cost* became nonzero (i.e., when any constraint had been violated). Second, in addition to the values in the original domains of the meta-variables, there is the possibility of an empty assignment ('ϵ') that serves to explicitly violate a constraint, and so the branching factor increases by exactly one. We refer to this decision branch as an *ϵ-relaxation*, as it performs an explicit relaxation of a weighted constraint. This latter modification, in combination with the meta-CSP search space employed in temporal reasoning, sets our algorithm apart from previous applications of weighted constraint satisfaction to classical CSPs.

Our experimental results, given in [3], show that the VDTP-based approach outperforms the previous approaches to finding utilitarian optimal solutions to DTPPs by several orders of magnitude.

6 Increased Expressive Power

A final research thrust for our project involved the identification of an important representational limitation of the temporal-constraint satisfaction formalisms and the development of new models that resolve this limitation. Specifically, prior work that had introduced temporal uncertainty into the formalism had all assumed that temporal un-

certainty is resolved in a specific way: the execution agent discovers the duration of a contingent link only at the exact moment of its conclusion (or equivalently, it discovers the time of a uncontrollable time point only at the time of its occurrence). This is not an accurate reflection of how such knowledge is obtained in many practical situations: one often learns the time of an initially uncertain event before its actual execution, and can use that knowledge to make informed decisions about the remainder of the plan. In addition, one may acquire only partial information about the time of an upcoming event. Finally, and perhaps surprisingly, certain varieties of uncertainty may exist even for problems that should be modeled entirely with controllable time points, something that again cannot be captured by previous formalisms.

We therefore generalized the problem of controllability in constraint-based temporal reasoning [2], developing three extensions to this formalism. The first introduces the notion of prior observability, in which the values of uncontrollable events become known prior to their actual occurrence. The second captures situations in which an observation event triggers the reduction of a contingent temporal interval, a form of partial observability that has not yet been captured in previous work. The third and final extension generalizes this notion to the requirement links, making it possible to express certain additional types of uncertainty. A further extension [1] added the capability of dealing with partial observability.

7 Experimental Validation

Although the work on this project was initially motivated by problems in cybersecurity and cyberdefense, it was difficult to obtain appropriate test sets on which to validate the algorithms. However, we were able to perform experimental validation using large-scale test sets from an alternative real-world domains.

In our first work on this topic [5], we tackled a stylized layout problem, rectangle packing. The problem of rectangle packing is one that has drawn attention from several diverse fields of computer science. For instance, in the context of scheduling, it can be used to represent scenarios where jobs require a fixed amount of time and resources, which compose the two dimensions of a single rectangle. The task of packing many such rectangles into an enclosing space, so as to minimize the width, height, or area of this space, allows for the minimization of makespan, resources needed, or total wasted resource. In VLSI design, rectangles represent actual physical modules that need to be placed in a spatial arrangement such that no two modules overlap.

Previous work had cast rectangle packing as a constraint satisfaction problem (CSP). In this formulation, a variable is created for each rectangle, whose legal values are the positions that rectangle could occupy without exceeding the boundaries of the enclosing space. In addition, there is a binary constraint between each pair of rectangles, requiring that they do not overlap. We developed an alternatie approach, casting the problem of optimally packing a set of rectangles with fixed orientations as a meta-CSP. In this formulation, we create a meta-variable for each pair of rectangles, whose values are the four pairwise relationships (i.e., above, below, left of, right of) that prevent that pair from overlapping. As such, commitment to the exact placement of any rectangle is not established until a consistent solution has been generated. By incorporating a

range of pruning techniques, including those mentioned earlier in this report, we obtained results demonstrating that our approach performs competitively compared to the previous state-of-the-art on a series of benchmarks. In addition, there is added flexibility in our approach that makes it much more amenable to extensions such as placement of rectangles without fixed orientation and scale up to very large rectangles, which are difficult for the fixed-placement formulation to handle efficiently.

We also tested our techniques in the real-world domain of circuit floorplanning legalization [4]. Floorplanning algorithms have traditionally underperformed experienced designers, even when relatively simple interconnect metrics are concerned. However, the sheer scale of modern systems on chip makes an all-manual design flow infeasible. We developed a new efficient automated approach to the floorplan repair problem, where a set of violated design constraints are satisfied by applying small changes to an existing rough floorplan. Such a floorplan can be produced by a human designer, by a scalable placement algorithm, or result from engineering adjustments to a pre-existing floorplan. In all cases, overlapping modules must be separated, and in some instances, modules may need to be repositioned to satisfy other requirements.

The algorithmic framework we developed uses the temporal constraint-satisfaction methods described above, adapted directly to the spatial domain. While capable of representing floorplans with or without overlapping modules, it can also support the outline of the core area, fixed module locations, region constraints, proximity and alignment constraints, etc. Instead of applying randomized local search in the hope of satisfying these constraints, we track all implications of imposed constraints and resolve violations by invoking gradual modifications to the floorplan. This approach proved to be extremeley effective: it completely eliminates overlaps from layouts produced by existing systems that produce rough floorplans (e.g., Capo 9.4, Feng Shui 5.1 and APlace 2.0) on IBM-HB benchmarks with hard blocks, typically requiring negligible runtime and increasing interconnect length by only several percent, significatly outperforming competing legalizers. Furthermore, we are able to generate legal solutions for these instances that surpass previously reported results in wirelength by an average of roughly 7%.

8 Publications Supported by the Project

1. M. D. Moffitt, "On the Partial Observability of Temporal Uncertainty," Proceedings of the 22nd National Conference on Artificial Intelligence, July 2007.

2. M. D. Moffitt and M. E. Pollack, "Generalizing Temporal Controllability," Proceedings of the 20th International Joint Conference on Artificial Intelligence, Jan. 2007.

3. M. D. Moffitt and M. E. Pollack, "Temporal Preference Optimization as Weighted Constraint Satisfaction," Proceedings of the 21st National Conference on Artificial Intelligence, July 2006.

4. M. D. Moffitt, A. N. Ng, I. L. Markov, and M. E. Pollack, "Constraint-Driven Floorplan Repair," Proceedings of the 43rd Design Automation Conference, July

2006.

5. M. D. Moffitt and M. E. Pollack, "Optimal Rectangle Packing: A Meta-CSP Approach," Proceedings of the 16th International Conference on AI Planning and Scheduling, June 2006. (Winner: Best Student Paper Award.)

6. M. D. Moffitt and M. E. Pollack, "Temporal Preference Optimization as Weighted Constraint Satisfaction," ICAPS-06 Workshop on Preferences and Soft Constraints in Planning, June 2006 (Preliminary version of [3]).)

7. H. Sheini, B. Peintner, K. Sakallah, and M. E. Pollack, "On Solving Soft Temporal Constraints using SAT Techniques," Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, Oct. 2005.

8. M. D. Moffitt and M. E. Pollack, "Applying Local Search to Disjunctive Temporal Problems," 19th International Joint Conference on Artificial Intelligence, Aug. 2005.

9. M. Liffiton, M. D. Moffitt, M. E. Pollack, and K. Sakallah, "Identifying Conflicts in Overconstrained Temporal Problems," 19th International Joint Conference on Artificial Intelligence, Aug. 2005.

10. B. Peintner, M. D. Moffit, and M. E. Pollack, "Solving Over-constrained DTPs with Preferences," 15th International Conference on Automated Planning and Scheduling, June 2005.

11. P. Schwartz and M. E. Pollack, "Two Approaches to Semi-Dynamic Disjunctive Temporal Problems," ICAPS Workshop on Constraint Programming for Planning and Scheduling, June 2005.

12. M. D. Moffitt, B. Peintner, and M. E. Pollack, "Augmenting Disjunctive Temporal Problems with Finite-Domain Constraints," 20th National Conference on Artificial Intelligence (AAAI), July 2005.

13. M. E. Pollack and I. Tsamardinos, "Efficiently Dispatching Plans Encoded as Simple Temporal Problems," in I. Vlahavas and D. Vrakas, editors, Intelligent Techniques for Planning, Idea Group, Inc. Hershey, PA, 2005.

14. B. Peintner and M. E. Pollack, "Anytime, Complete Algorithm for Finding Utilitarian Optimal Solutions to STPPs," 20th National Conference on Artificial Intelligence (AAAI), July 2005.

15. M. D. Moffitt and M. E. Pollack, "Partial Constraint Satisfaction of Disjunctive Temporal Problems," 18th International Florida AI Research Symposium (FLAIRS), May 2005. 2nd Place Best Paper Award.

16. P. Schwartz and M. E. Pollack, "Planning with Disjunctive Temporal Constraints," ICAPS-04 Workshop on Integrating Planning into Scheduling, June, 2004.

17. B. Peintner and M. E. Pollack, "Low-Cost Addition of Preferences to DTPs and TCSPs," 19th National Conference on Artificial Intelligence, July, 2004.

9 Personnel Supported by the Project

This project provided support for the Principal Investigator, Dr. Martha Pollack, as well as two graduate students: Michael Moffitt and Peter Schwartz. Moffitt completed his Ph.D. degree during the period of performance (May 2007); Schwartz is scheduled to defend his Ph.D. thesis in Aug. 2007. A third student, Jacob Balazer, was supported for one semester early in the first year of the project.

10 Transitions, Interactions, and Honors

- Tutorial Presentation given by M. Pollack, "Temporal and Resource Reasoning for Planning, Scheduling, and Execution", all day tutorial at the 21st National Conference on Artificial Intelligence (AAAI), jointly with Dr. Nicola Muscettola, Lockheed Martin, July 2006.
- Tutorial Presentation given by M. Pollack, "Temporal and Resource Reasoning for Planning, Scheduling, and Execution", all day tutorial at the15th International Conference on AI Planning and Scheduing, jointly with Dr. Nicola Muscettola, NASA Ames, June 2005.
- Certain of the reasoning algorithms and software developed within this project have been adopted for use in the DARPA PAL (Perceptive Agent that Learns) program, a large, five year, multi-million dollar showcase project aimed at developing an enduring personalized cognitive assistant. Pollack has a subcontract through SRI, International, one of the prime contractors on the PAL project, to help incorporate several of the plan management technologies into the core of the system being developed.
- Visit to AFRL by M. Pollack, Dec. 2004, to meet with Dr. Chet Maciag to discuss cybersecurity and cyberdefense applications of the work, followed up with email discussions with Mr. Chad Korose, Naval Reservist working with Dr. Joseph Giordano.
- M. Pollack was elected to the CRA (Computing Research Association) Board of Directors, 2007-2009.
- M. Moffitt won the IBM 2007 Josef Raviv Memorial Postdoctoral Fellowship.
- M. Moffitt won 1st Place in the ISPD (Internatinonal Symposium on Physical Design) 2007 Global Routing Contest for his MaizeRouter algorithm.
- M. Pollack was appointed as a member of the National Science Foundation's Computer and Information Science and Engineering (CISE) Advisory Committee, 2006-2008.
- M. Pollack gave the following invited talks. Note that while the topic of these talks was on Assistive Technology, a number of the algorithms we used in our assistive technology projects grew out of those developed in this AFOSR effort:

- "Intelligent Assistive Technology: The Present and the Future": University of Southern California Department of Computer Science Distinguished Lecture Series, March 2007; Monterey Bay Aquarium Research Institute Seminar, March 2007; 20th International Joint Conference on Artificial Intelligence Invited Plenary Talk, January 2007; University of Massachusetts Distinguished Lecture Series, November 2006.
- "Intelligent Technology for Adaptive Aging": Plenary Talk at the 18th International Florida AI Research Symposium, May 2005; Harvard University Computer Science Colloquium, April 2005; Columbia University Computer Science Department 25th Anniversary Lecture Series, Feb. 2005; University of Wisconsin Dept. of Biostatistics and Medical Informatics, Feb. 2005; Invited Plenary Talk at the 19th International Conference on Artificial Intelligence, July 2004.